Working with Rasters and Imagery using Python

Jing Li

2019 ESRI DEVELOPER SUMMIT
Palm Springs, CA

Imagery Data in ArcGIS

- **ArcGIS supports imagery data management from various sources**
  - **Over 100 native image formats**
  - **Over 60 different sensors supported**
    - **Satellite**
    - **Aerial cameras**
    - **UAV/UAS cameras**
    - **Multidimensional data NetCDF/GRIB/HDF**

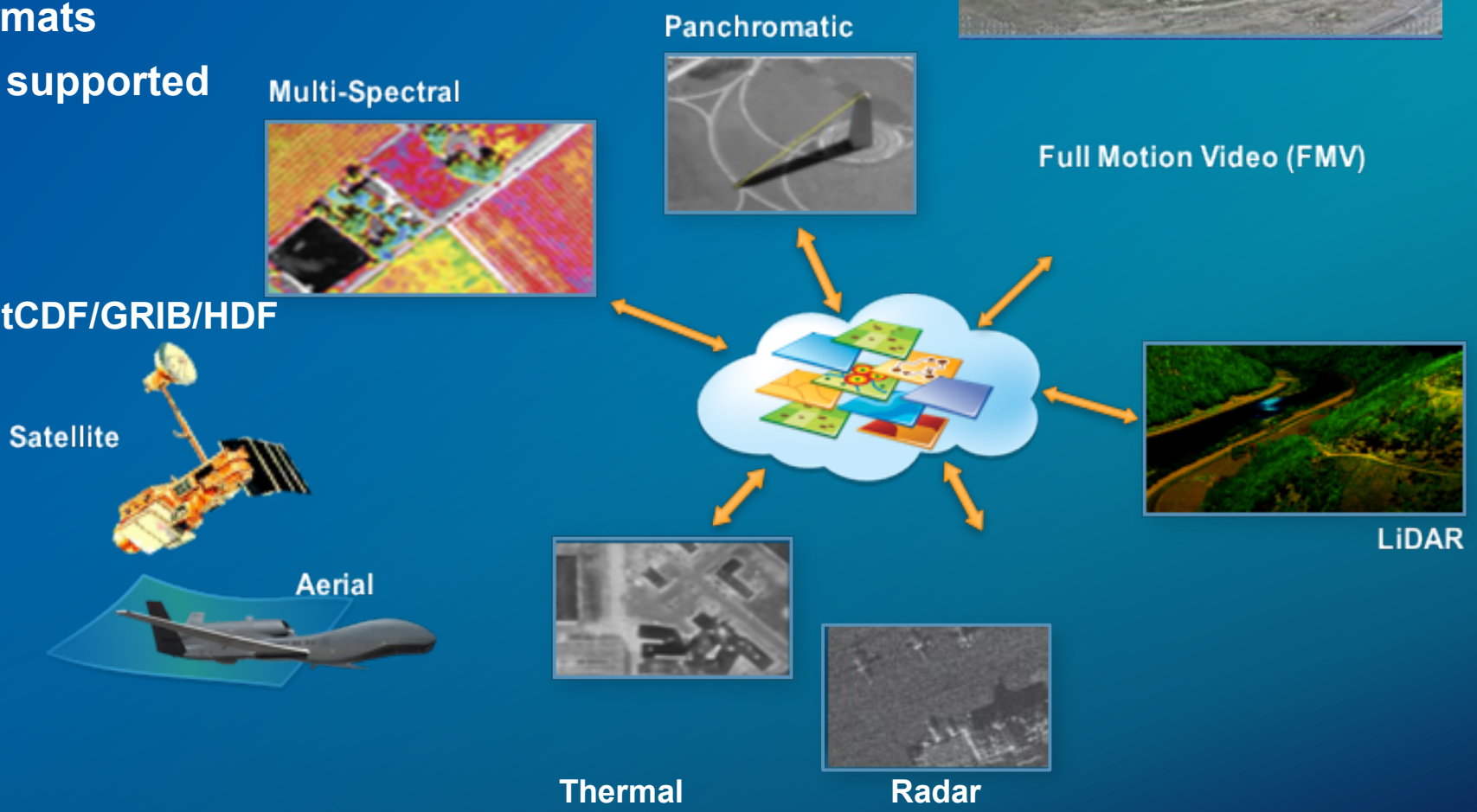
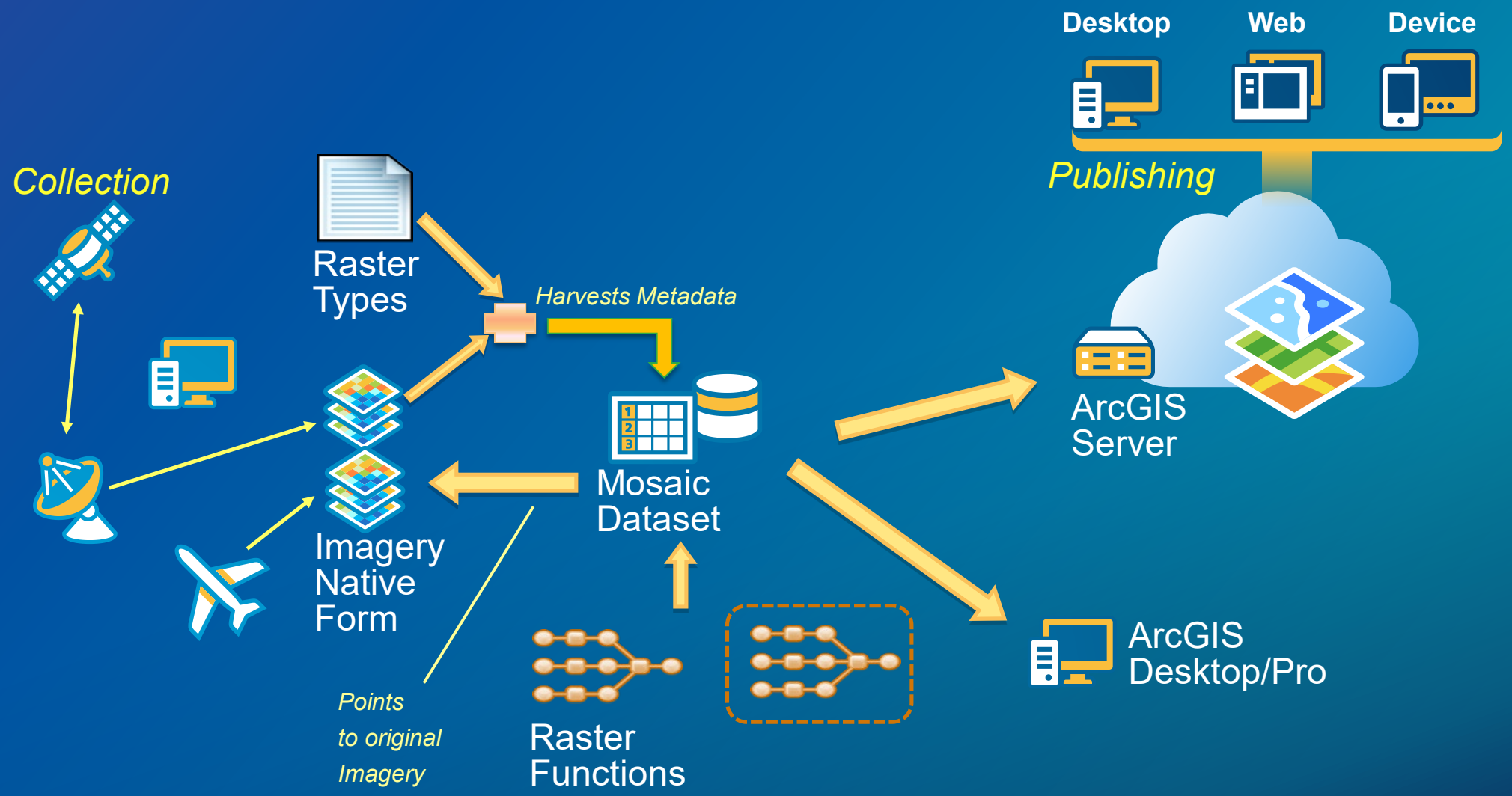
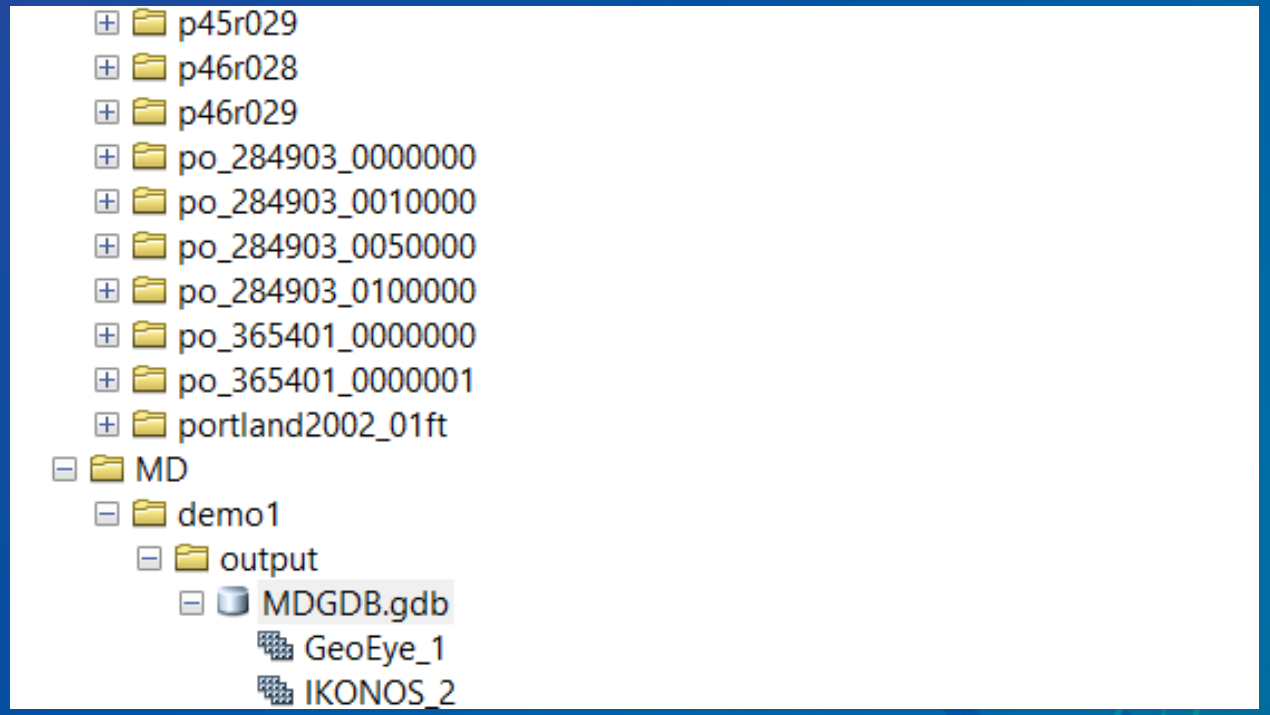


Image Management using Mosaic Dataset

- **ArcGIS provides image collection management solution through Mosaic Dataset**
  - **Create mosaic dataset in supported geodatabase**
    - **File geodatabase**
    - **Enterprise geodatabase (Oracle/PostgreSQL/SQL Server e.t.c)**
  - **Providing a catalog view of image collection**
  - **Referencing original image source**
  - **Dynamic mosaicking**
  - **Support on-the-fly raster function**
  - **Can be shared as image service through Image Server**

Automate Imagery Management and Sharing Workflow





Automate Mosaic Dataset Creation

Explore Imagery Properties

- **Look up raster data in your workspace**

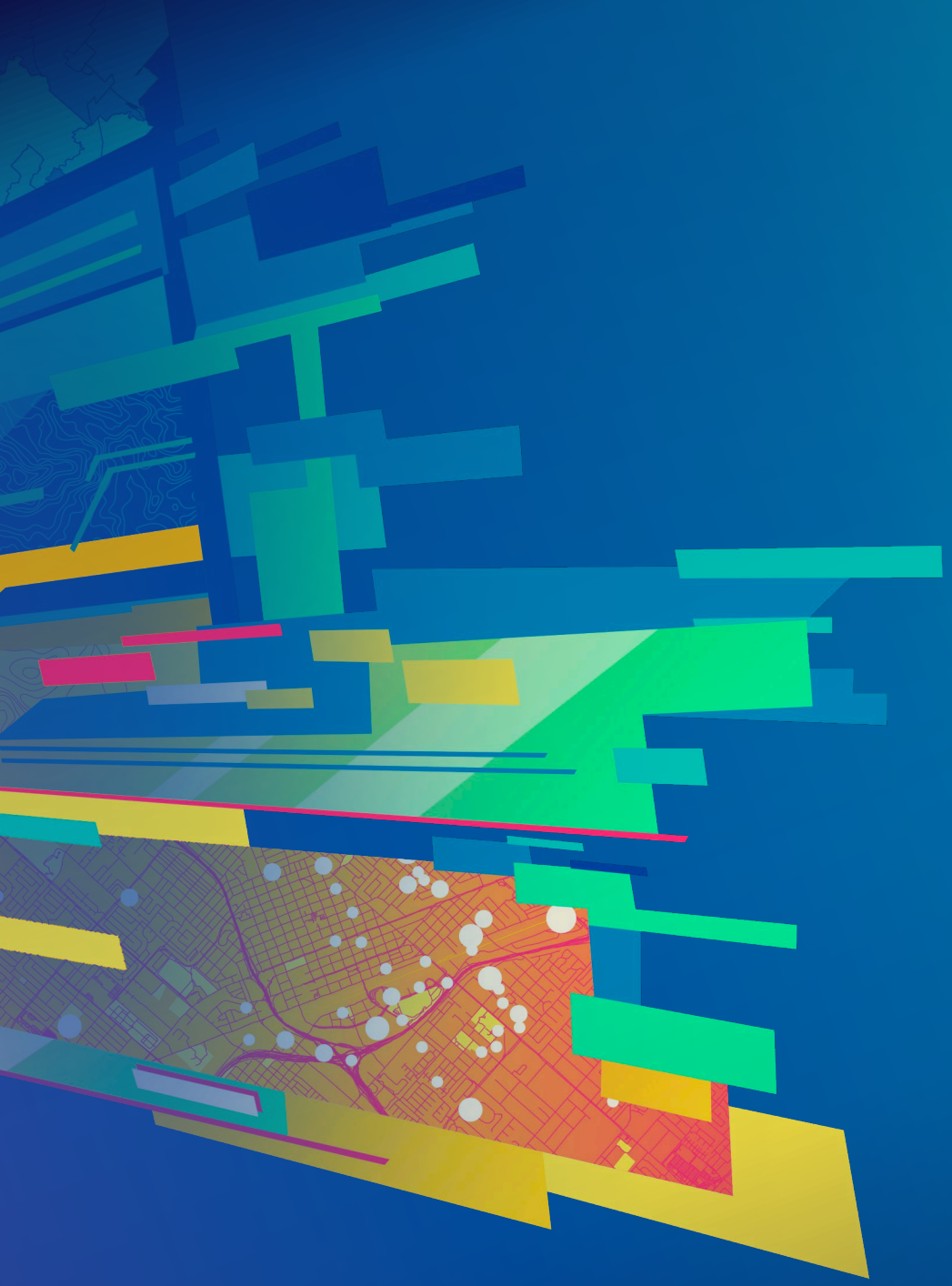
```
arcpy.env.workspace = workspace
rasterds = arcpy.ListRasters()
for raster in rasterds:
    yield os.path.join(workspace, raster)
```

- **Get raster property**

```
#get the sensorName property from the raster dataset
sensorNameResult = arcpy.GetRasterProperties_management(
    raster, "SENSORNAME")
```

- **Use correct raster type to load data into mosaic dataset**

```
# create mosaic dataset
arcpy.env.overwriteOutput = 1
arcpy.CreateMosaicDataset_management(gdbName, mdName, "54004")

# load data for this raster type
arcpy.AddRastersToMosaicDataset_management(
    os.path.join(gdbName, mdName), rasterType, indir)
```

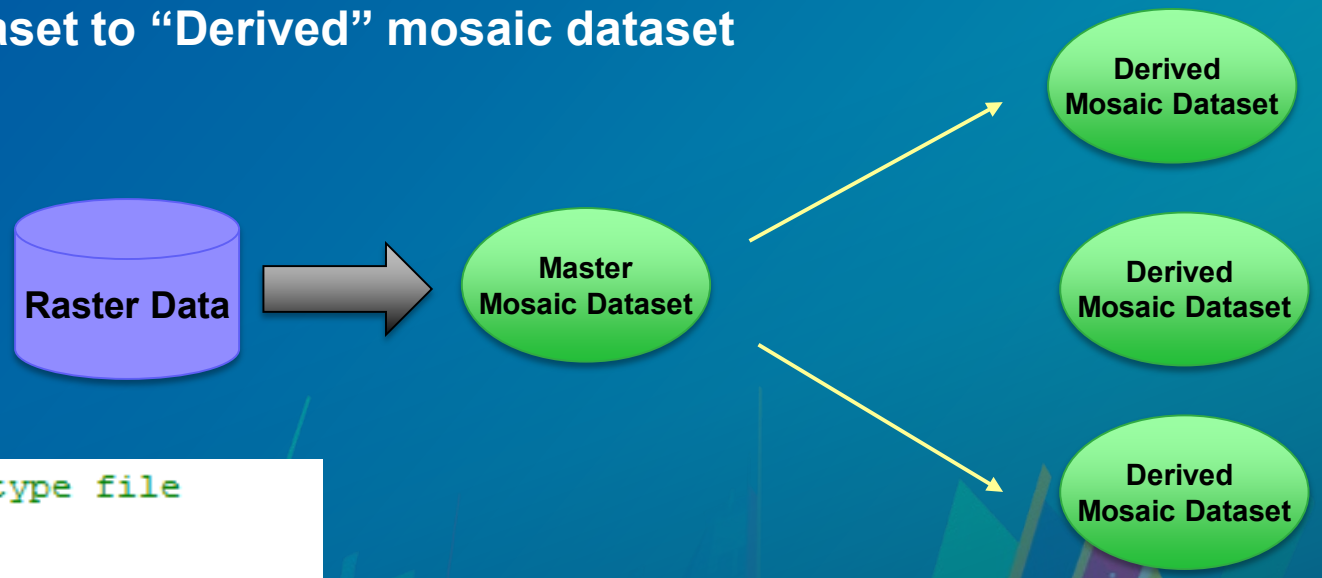


Demo

Image Discovery and Mosaic Dataset Creation

Create Derived Mosaic Dataset

- **Create subsets of your imagery collection for different projects**
  - **Use one “Master” mosaic dataset for all imagery data**
  - **Add subset of “Master” mosaic dataset to “Derived” mosaic dataset**



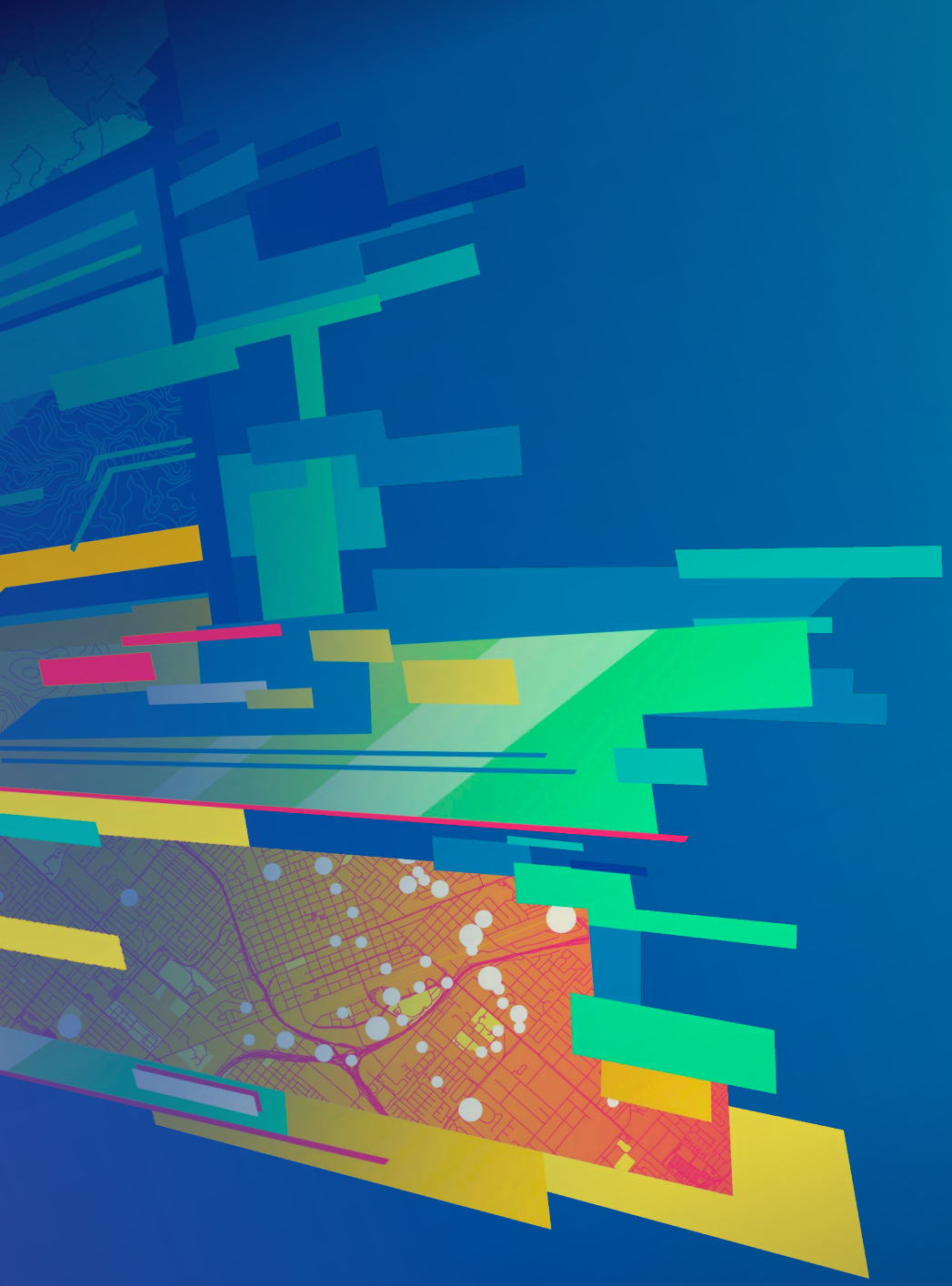

```
# Add rasters using the Table raster type file
# to create derived mosaic dataset
mdpath = os.path.join(
    arcpy.env.workspace, "FGDB.gdb/Geoeye1nIKONOS")
inputgeoeye = r"e:\MDGDB.gdb\sensorType2"
arcpy.AddRastersToMosaicDataset_management(
    mdpath, "Table", inputgeoeye)
```

Customize Mosaic Dataset

- **Customize raster type settings**
  - **Use Aux input parameter in “Add Rasters to Mosaic Dataset” tool**
  - **Edit raster type *art.xml file**
- **Use arcpy.da cursor to access mosaic dataset footprints table**
  - **Read raster object from Raster field**
  - **Read/write field values**

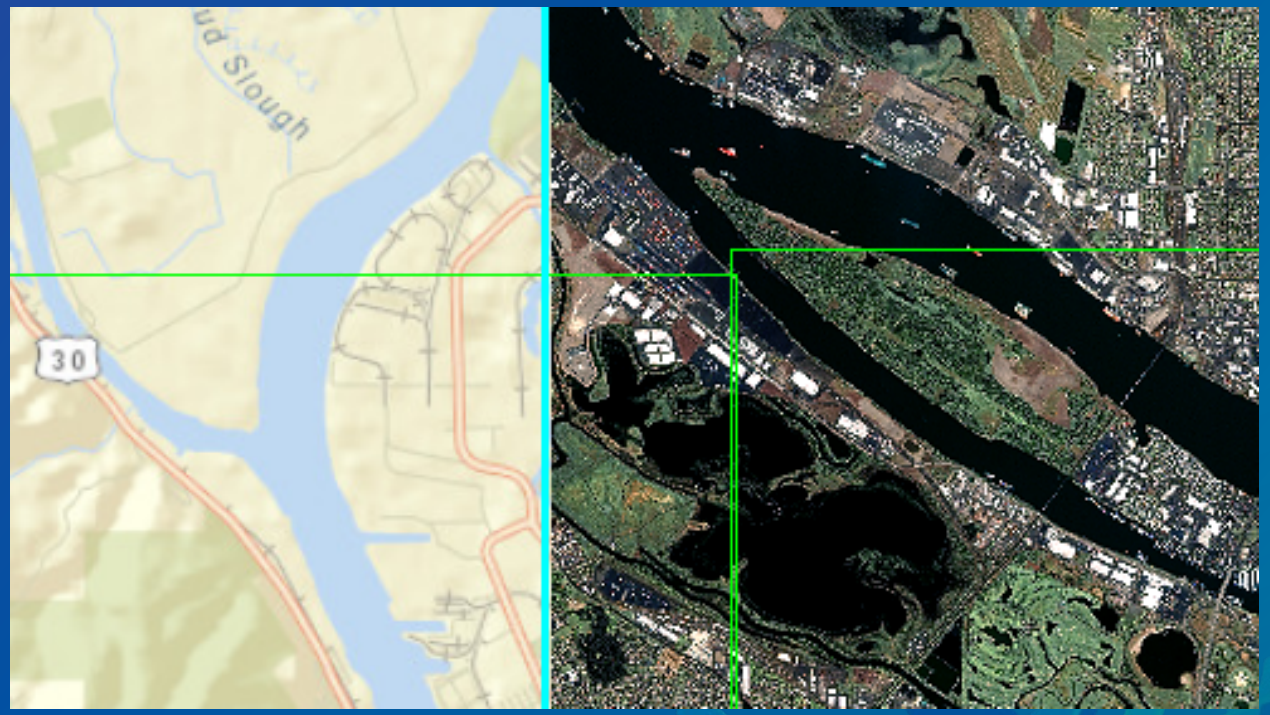
```
rasfields = ["OBJECTID", "Raster"]
with arcpy.da.SearchCursor(mdpath, rasfields) as rcursor:
    for row in rcursor:
        #Create Raster object directly from cursor
        bluemin = arcpy.GetRasterProperties_management(
            row[1], "WAVELENGTH", "BLUE")
```

- **Use Mosaic Dataset Geoprocessing tool set through arcpy**
  - **arcpy.AlterMosaicDatasetSchema_management()**
  - **arcpy.DefineOverviews_management()**
  - **....**



Demo

Creating Derived Mosaic Datasets



Automate Imagery Sharing

Automate image service sharing

- **Create publisher server connection file**

```
conType = "PUBLISH_GIS_SERVICES"
folderPath = os.path.join(os.getcwd(), "output")
fileName = serverName + "_publisher.ags"
serverURL = "http://"+serverName+":6080/arcgis"
serverType = "ARCGIS_SERVER"

arcpy.mapping.CreateGISServerConnectionFile(
    conType, folderPath, fileName, serverURL, serverType,
    username=userName, password=passWord)
```

- **Create image service definition draft**

```
sddraftPath = os.path.join(
    folderPath, serviceName+".sddraft")
arcpy.CreateImageSDDraft(
    mdPath, sddraftPath, serviceName, "ARCGIS_SERVER",
    copy_data_to_server=False)
```

Automate image service sharing

- **Modify service setting by editing *.sddraft file**
- **Analyze image service definition draft before publishing**

```
analysis = arcpy.mapping.AnalyzeForSD(sddraftPath)

for key in ('messages', 'warnings', 'errors'):
    print "----" + key.upper() + "---"
    vars = analysis[key]
    for ((message, code), data) in vars.iteritems():
        print "    ", message, " (CODE %i)" % code
```


```
  <OnServerName/>
- <Configurations xsi:type="typens:ArrayOfSVCConfiguration">
  - <SVCConfiguration xsi:type="typens:SVCConfiguration">
      <ID>63B5AEEC-4B1A-4CA4-B434-E1A631934596</ID>
      <Name>DS2019</Name>
      <TypeName>ImageServer</TypeName>
      <ResourceID>{05858694-F6F8-4CA6-BF50-B95F14B06EBC}</ResourceID>
      <ServiceFolder/>
      <DataFolder/>
    - <Definition xsi:type="typens:SVCConfigurationDefinition">
        <Description/>
      - <ConfigurationProperties xsi:type="typens:PropertySet">
        - <PropertyArray xsi:type="typens:ArrayOfPropertySetProperty">
            + <PropertySetProperty xsi:type="typens:PropertySetProperty">
            + <PropertySetProperty xsi:type="typens:PropertySetProperty">
            + <PropertySetProperty xsi:type="typens:PropertySetProperty">
            + <PropertySetProperty xsi:type="typens:PropertySetProperty">
            + <PropertySetProperty xsi:type="typens:PropertySetProperty">
            - <PropertySetProperty xsi:type="typens:PropertySetProperty">
                <Key>rasterTypes</Key>
                <Value xsi:type="xs:string">Raster Dataset</Value>
              </PropertySetProperty>
            - <PropertySetProperty xsi:type="typens:PropertySetProperty">
                <Key>rasterFunctions</Key>
                <Value xsi:type="xs:string">C:\DevSummit\DevSummit2019\Code\demo3\Portland_NDVI.rft.xml</Value>
              </PropertySetProperty>
            - <PropertySetProperty xsi:type="typens:PropertySetProperty">
                <Key>minInstances</Key>
                <Value xsi:type="xs:string">1</Value>
              </PropertySetProperty>
```

Automate image service sharing

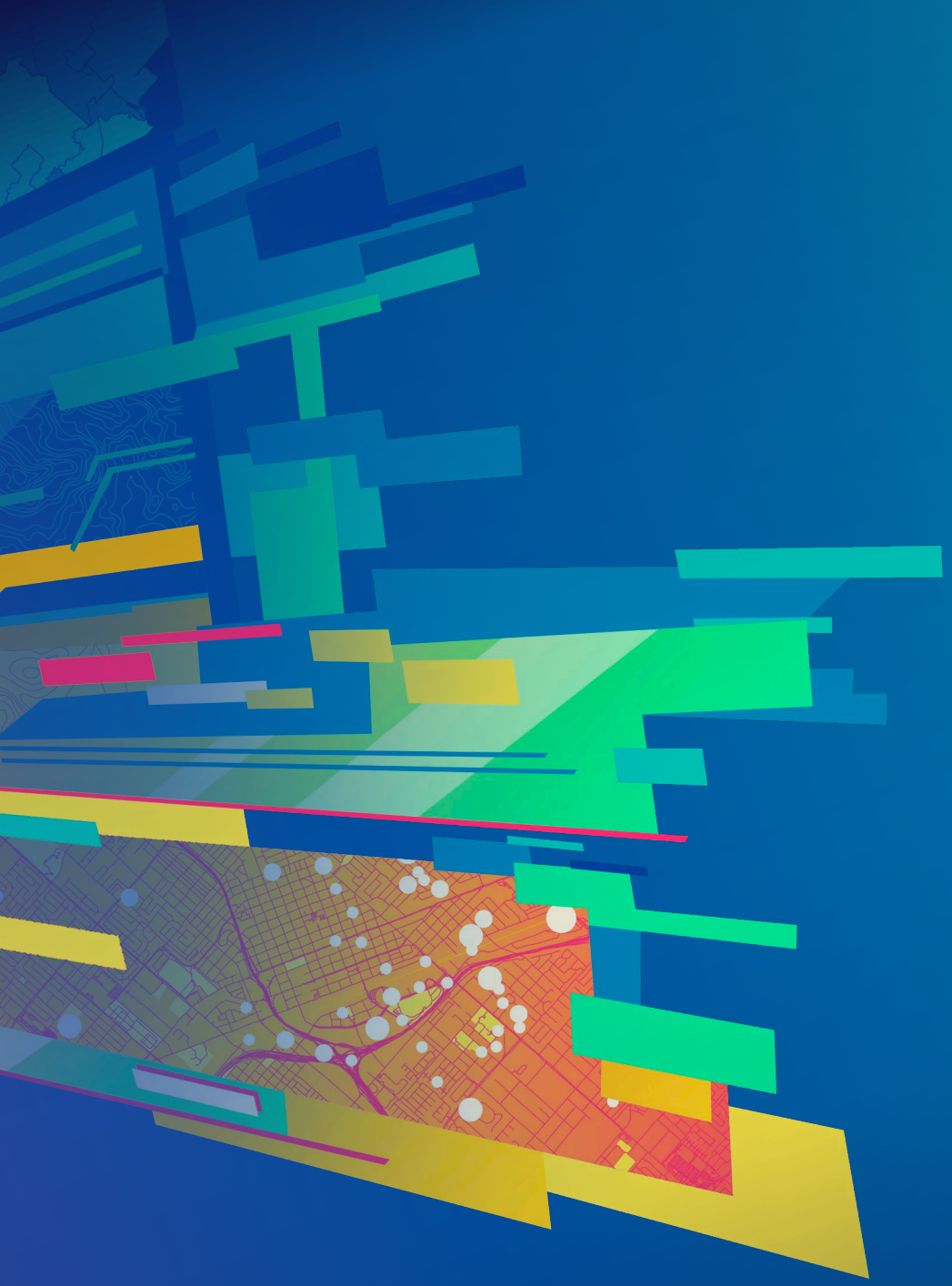
- **Stage *.sddraft to *.sd (Service Definition)**

```
sdPath = sddraftPath.replace(".sddraft", ".sd")
arcpy.StageService_server(sddraftPath, sdPath)
```

- **Upload service definition to server**

```
arcpy.UploadServiceDefinition_server(sdPath, connectionfile)
```

- **<u>Be aware of different publishing mode</u>**
  - **Publish image service by value – data will be copied**
  - **Publish by reference – data location needs to be registered as server data store**



Demo

Publishing Image Services

Use Image Service REST API in Python

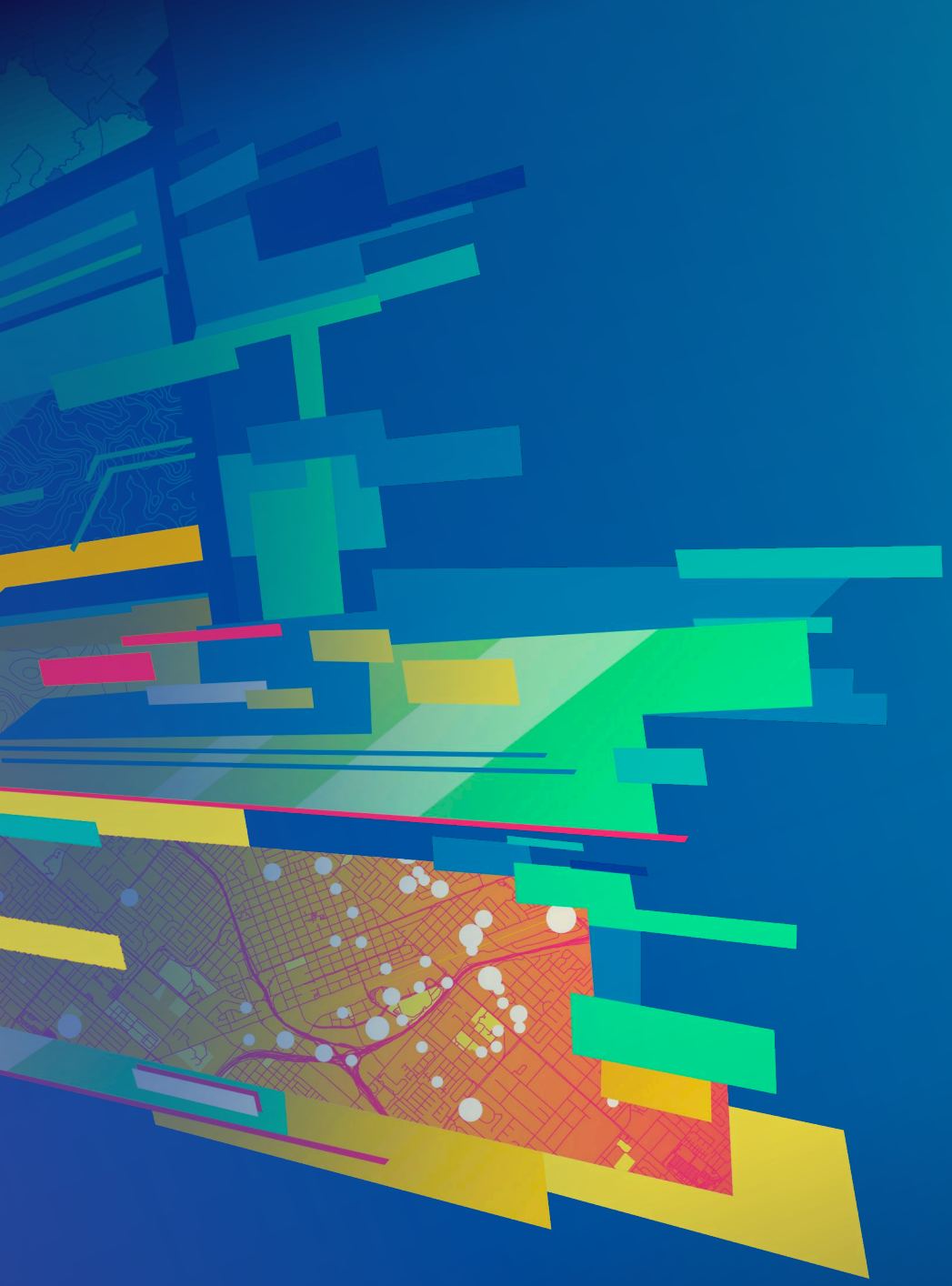
Image Service REST API

- Get general service information
- Query individual raster item
- Export Image
  - Define geometry
  - Define mosaic rule
    - LockRaster to export from specific item
  - Support compression
  - Request different rendering rules
  - Export format
- Many more… http://esriurl.com/isrest

Use Image Service REST API in Python

- **Python has many modules can be used for RESTful requests**
  - **requests (now comes with ArcGIS Python)**
  - **urllib2 etc.**
- **Example:**
  - **Custom Geoprocessing tool to clip and export**
  - **Make selection**
  - **Persist on-the-fly processing**
  - **Determine image order using mosaic rule**

```python
post_data = {
    "bbox": str(bbox).lstrip("[").rstrip("]"),
    "size": str(width) + "," + str(height),
    "imageSR": json.dumps(isprj),
    "bboxSR": json.dumps(isprj),
    "format": "tiff",
    "compression": "LZ77",
    "pixelType": pixtype,
    "mosaicRule": json.dumps({'mosaicMethod': 'esriMosaicLockRaster', 'lockRasterIds': [str(itemoid)]}),
    "renderingRule": json.dumps(rrule),
    "f": "json"
}

serviceURL = serviceURL.replace("arcgis/services", "arcgis/rest/services") + "/exportImage"

# arcpy.AddMessage("exportImage REST URL: {}".format(serviceURL))
# arcpy.AddMessage("exportImage REST parameters: {}".format(json.dumps(post_data)))
#Read response from server and find image url
arcpy.AddMessage("Requesting to export image id = " + str(itemoid))

req = requests.post(serviceURL, data=post_data, verify=False)

jsondict = req.json()
imageurl = jsondict["href"]

# arcpy.AddMessage("exportImage response: {}".format(json.dumps(jsondict)))
# Retrive image url from server
file_name = os.path.join(outputws, "mdimage" + str(itemoid) + ".tif")
if arcpy.Exists(file_name):
    arcpy.Delete_management(file_name)

with open(file_name, 'wb') as f:
    f.write(requests.post(imageurl).content)
arcpy.AddMessage("Successfully download image id = " + str(itemoid))
```



Demo

Clipping and Exporting Image Services

Thank you!

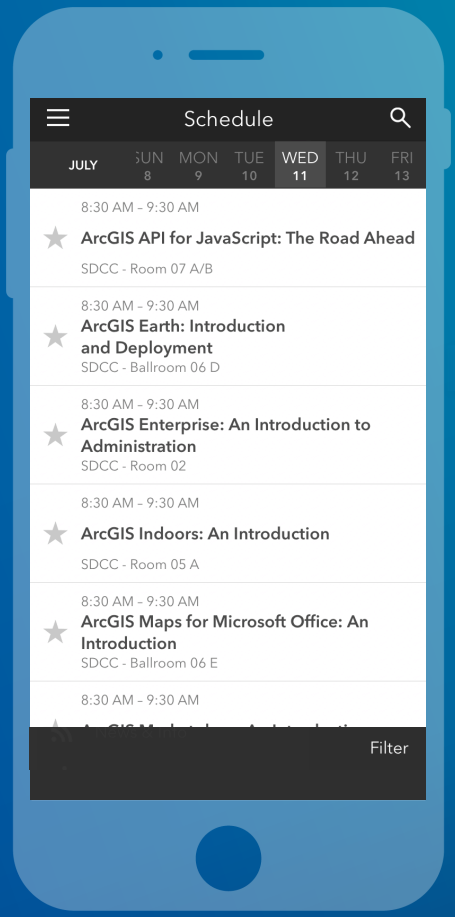
esri

THE SCIENCE OF WHERE

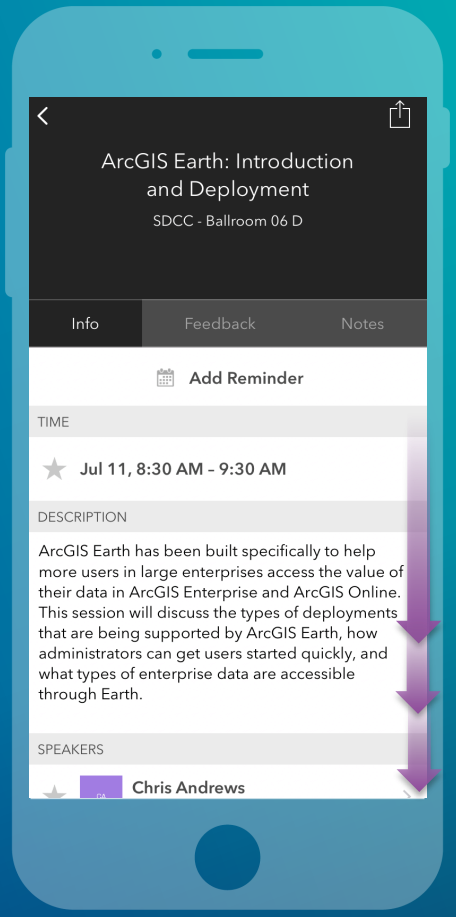
Please Take Our Survey on the App

Download the Esri Events app and find your event

Select the session you attended



Scroll down to find the feedback section



Complete answers and select “Submit”